Artificial Intelligence Programming
Bayesian Networks

Chris Brooks

Department of Computer Science
University of San Francisco
Previously, we talked about rule-based systems that can draw logical inferences.

- $Hungry(Homer) \Rightarrow GoesTo(Homer, Quickie - mart)$
- $Hungry(Homer)$
- $GoesTo(Homer, Quickie - mart)$

We would like to extend this sort of operation to worlds with uncertainty.

- $P(Hungry(Homer)) = 0.98$
- $P(GoesTo(Homer, Quickie - mart) | Hungry(Homer)) = 0.5$
- $P(GoesTo(Homer, Quickie - mart)) = ?$
In general, we can use Bayes’ Rule to do this.

The problem is in working with the joint probability distribution.

- Exponentially large table.

We need a data structure that captures the fact that many variables do not influence each other.

- For example, the color of Bart’s hat does not influence whether Homer is hungry.

We call this structure a Bayesian network (or a belief network)
A Bayesian network is a directed graph in which each node is annotated with probability information. A network has:

- A set of random variables that correspond to the nodes in the network.
- A set of directed edges or arrows connecting nodes. These represent influence. In there is an arrow from $X$ to $Y$, then $X$ is the parent of $Y$.
- Each node keeps a conditional probability distribution indicating the probability of each value it can take, conditional on its parents values.
- No cycles. (it is a directed acyclic graph)

The topology of the network specifies what variables directly influence other variables. (conditional independence relationships).
19-3: Burglary example

- Two neighbors will call when they hear your alarm.
  - John sometimes overreacts
  - Mary sometimes misses the alarm.

- Two things can set off the alarm
  - Earthquake
  - Burglary

- Given who has called, what’s the probability of a burglary?
Each node has a conditional probability table.

This gives the probability of each value of that node, given its parents’ values.

These sum to 1.

Nodes with no parents just contain priors.
6 Notice that we don’t need to have nodes for all the reasons why Mary might not call.
   △ A probabilistic approach lets us summarize this information in $\neg M$

6 This allows a small agent to deal with large worlds that have a large number of possibly uncertain outcomes.

6 How would we handle this with logic?
Recall that the full joint distribution allows us to calculate the probability of any variable, given all the others.

Independent events can be separated into separate tables.

These are the CPTs seen in the Bayesian network.

Therefore, we can use this info to perform computations.

\[ P(x_1, x_2, \ldots, x_n) = \Pi P(x_i | \text{parents}(x_i)) \]

\[ P(A \land \neg E \land \neg B \land J \land M) = \]
\[ P(J | A) P(M | A) P(A | \neg B \land \neg E) P(\neg B) P(\neg E) = \]
\[ 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998 = 0.00062 \]
There are often several ways to construct a Bayesian network.

The knowledge engineer needs to discover *conditional independence* relationships.

Parents of a node should be those variables that directly influence its value.
- JohnCalls is influenced by Earthquake, but not directly.
- John and Mary calling don’t influence each other.

Formally, we believe that:

\[ P(\text{MaryCalls} | \text{JohnCalls}, \text{Alarm}, \text{Earthquake}, \text{Burglary}) = P(\text{MaryCalls} | \text{Alarm}) \]
Bayesian networks allow for a more compact representation of the domain

- Redundant information is removed.

Example: Say we have 30 nodes, each with 5 parents.

Each CPT will contain $2^5 = 32$ numbers Total: 960.

Joint: $2^{30}$ entries, nearly all redundant.
Begin with root causes
Then add the variables they directly influence.
Then add their direct children.
This is called a causal model
  Reasoning in terms of cause and effect
Estimates are much easier to come up with this way.
We could try to build from effect to cause, but the network would be more complex, and the CPTs hard to estimate. $P(E|B) = ?$
Recall that conditional independence means that two variables are independent of each other, given the observation of a third variable.

\[ P(a \land b | c) = P(a | c) P(b | c) \]

A node is conditionally independent of its nondescendants, given its parents.

- Given Alarm, JohnCalls is independent of Burglary and Earthquake.

A node is conditionally independent of all other nodes, given its parents, children, and siblings (the children’s other parents).

- Burglary is independent of JohnCalls given Alarm and Earthquake.
In most cases, we’ll want to use a Bayesian network to tell us posterior probabilities.

We observe a variable - how do other variables change?

We can distinguish between query variables (things we want to know about) and evidence variables (things we can observe).

There are also hidden variables, which influence query variables but are not directly observable.

JohnCalls and MaryCalls are evidence, Earthquake and Burglary are queries, and Alarm is hidden.
Recall that the probability
\[ P(X|e) = \alpha P(X, e) = \alpha \sum P(X, e, y) \]
where \( y \) are hidden variables.

This is equivalent to summing within the joint probability distribution.

Consider \( P(Burglary|JohnCalls, MaryCalls) = \alpha P(Burglary \land JohnCalls \land MaryCalls) \)

This is \( \alpha \sum_E \sum_A P(B, E, A, J, M) \)

For \( B = true \), we must calculate:
\[ P(B|J, M) = \alpha \sum_E \sum_A P(B)P(E)P(A|B, E)P(J|A)P(M|A) \]
Problem- a network with $n$ nodes will require $O(n2^n)$ computations.

We can simplify by bringing the priors outside the summation.

\[ \alpha P(B) \sum_E P(E) \sum_A P(A|B, E)P(J|A)P(M|A) \]

Still requires $O(2^n)$ calculations.
19-15: Enumeration

This tree shows the computations needed to determine $P(B|J, M)$.

Notice that there are lots of repeated computations in here.

By eliminating repeated computations, we can speed things up.
We can reduce the number of calculations by caching results and reusing them.

Evaluate the expression from right-to-left.

Summations are done only for nodes influenced by the variable being summed.

Consider
\[ P(B|j, m) = \alpha P(B) \sum_e P(e) \sum_a P(a|B, e) P(j|a) P(m|a) \]

The equation has been separated into factors.

Once we compute \( P(M|a) \) and \( P(M|\neg a) \), we cache those results in a matrix \( f_M \).

Similarly for \( P(J|a) \) and \( P(J|\neg a) \) stored in \( f_J \).
Variable Elimination

$P(A|B, E)$ will result in a $2 \times 2 \times 2$ matrix, stored in $F_A$.

We then need to compute the sum over the different possible values of $A$

This sum is $\sum_a f_A(A, B, E)f_J(A)f_M(A)$

We can process $E$ the same way.

In essence, we’re doing dynamic programming here, and exploiting the same memoization process.

Complexity

- Polytrees: (one undirected path between any two nodes) - linear in the number of nodes.
- Multiply-connected nodes: Exponential.
Many techniques have been developed to allow Bayesian networks to scale to hundreds of nodes.

**Clustering**
- Nodes are joined together to make the network into a polytree.
- CPT within the node grows, but network structure is simplified.

**Approximating inference**
- Monte Carlo sampling is used to estimate conditional probabilities.
19-19: Monte Carlo sampling example

Start at the top of the network and select a random sample. (say it’s true).

Draw a random sample from its children. conditioned on true.

\[ P(Sprinkler|Cloudy = true) = <0.1, 0.9> \text{. Say we select False} \]

\[ P(Rain|Cloudy = true) = <0.8, 0.2> \text{. Say we select True} \]

\[ P(WetGrass|Sprinkler = \neg false, Rain = true) = <0.9, 0.1> \text{. Say we select true.} \]

This gives us a sample for \(<\text{cloudy, } \neg \text{Sprinkler, Rain, WetGrass}>\)

As we increase the number of samples, this provides an estimate of \(P(\text{cloudy } \neg \text{Sprinkler, Rain, WetGrass})\).

In this way, we choose the query we are interested in and then sample the network "enough" times to determine the probability of that event occurring.
Other Approaches to Uncertainty

- Default reasoning
  - Logical
  - “Unless we know otherwise, A follows from B”
- Dempster-Shafer
  - Incorporates uncertainty about probabilities
- Fuzzy logic
  - Allows “partially true” statements.
Diagnosis (widely used in Microsoft’s products)
Medical diagnosis
Spam filtering
Expert systems applications (plant control, monitoring)
Robotic control